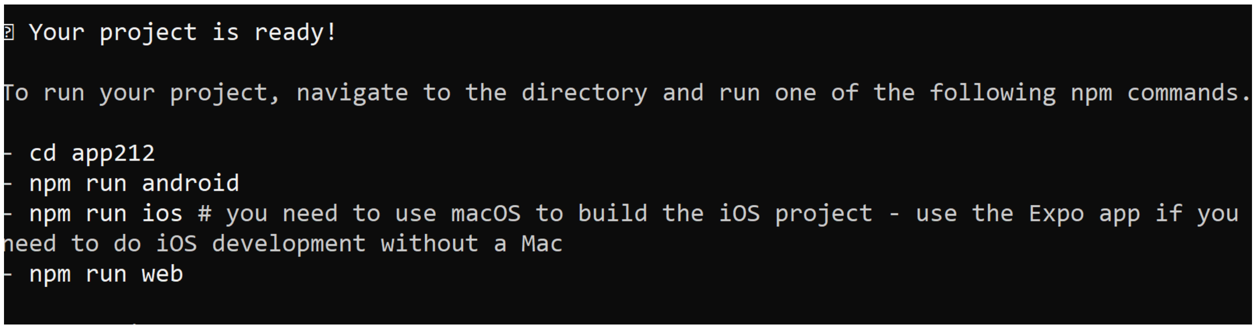
1. **npx create-expo-app@latest --template**

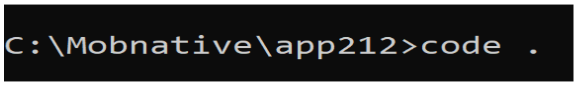
Choose Default

√ Choose a template: » Default

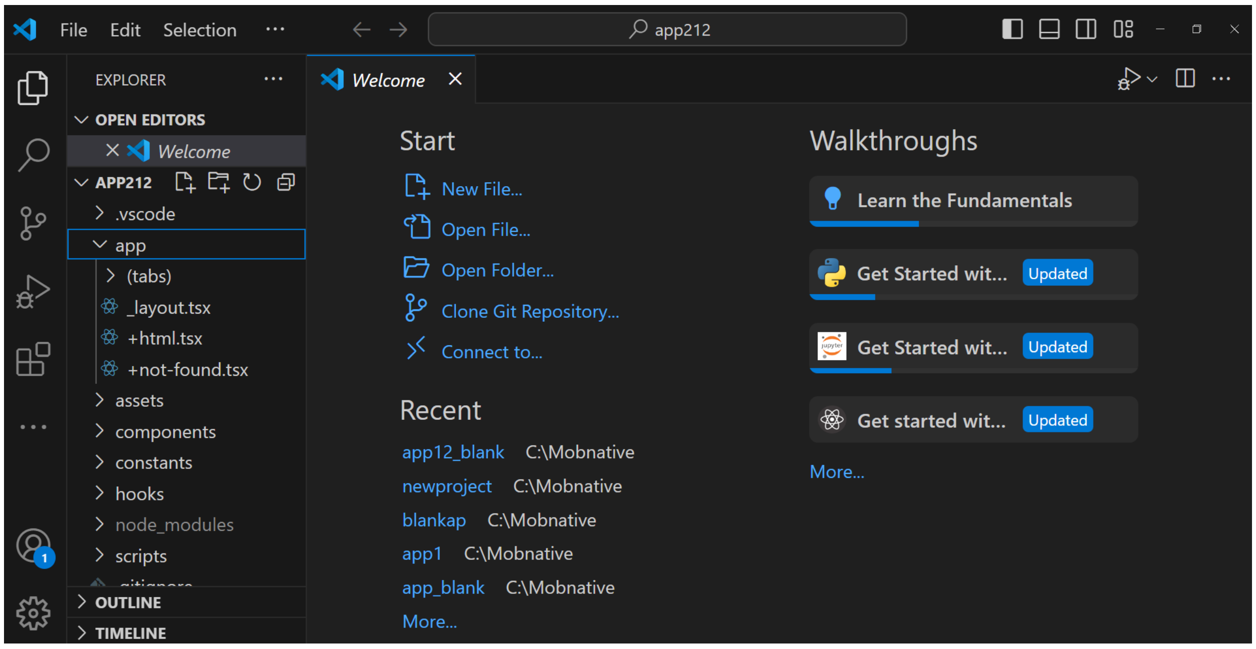
? What is your app named? » my-app

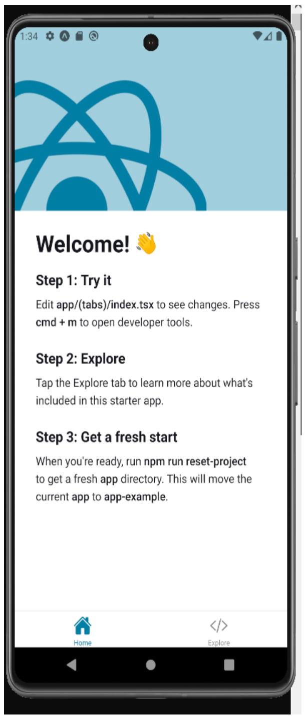


```
? Your project is ready!

To run your project, navigate to the directory and run one of the following npm commands.

- cd app212
- npm run android
- npm run ios # you need to use macOS to build the iOS project - use the Expo app if you
need to do iOS development without a Mac
- npm run web
```



```
C:\Mobnative\app212>code .
```





Welcome! 👋

Step 1: Try it

Edit **app/(tabs)/index.tsx** to see changes. Press **cmd + m** to open developer tools.

Step 2: Explore

Tap the Explore tab to learn more about what's included in this starter app.

Step 3: Get a fresh start

When you're ready, run **npm run reset-project** to get a fresh **app** directory. This will move the current **app** to **app-example**.

2. **npx create-expo-app StickerSmash --template blank && cd StickerSmash**

This command will create a new directory for the project with the name **StickerSmash**. The create-expo-app has a --template option, which we can use to create and set up a new project with different pre-installed libraries. In this case, we're using the blank which installs the minimum required libraries. We'll continue to add more libraries throughout this tutorial when needed.

Install dependencies
To run the project on the web, we need to install the following dependencies that will help to run the project on the web:

npx expo install react-dom react-native-web @expo/metro-runtime

Manual installation
Follow the steps below if you have a project that was previously created with Expo but does not have Expo Router library installed.

1. **Install dependencies**

You'll need to install the following dependencies:
npx expo install expo-router react-native-safe-area-context react-native-screens expo-linking expo-constants expo-status-bar

2. Setup entry point
For the property main, use the expo-router/entry as its value in the package.json. The initial client file is app/_layout.js.

```
{
  "main": "expo-router/entry"
```

3.Modify project configuration

Add a deep linking scheme in your app config:

```
{
  "scheme": "your-app-scheme"
}
```

4.Clear bundler cache
After updating the Babel config file, run the following command to clear the bundler cache:
npx expo start -c

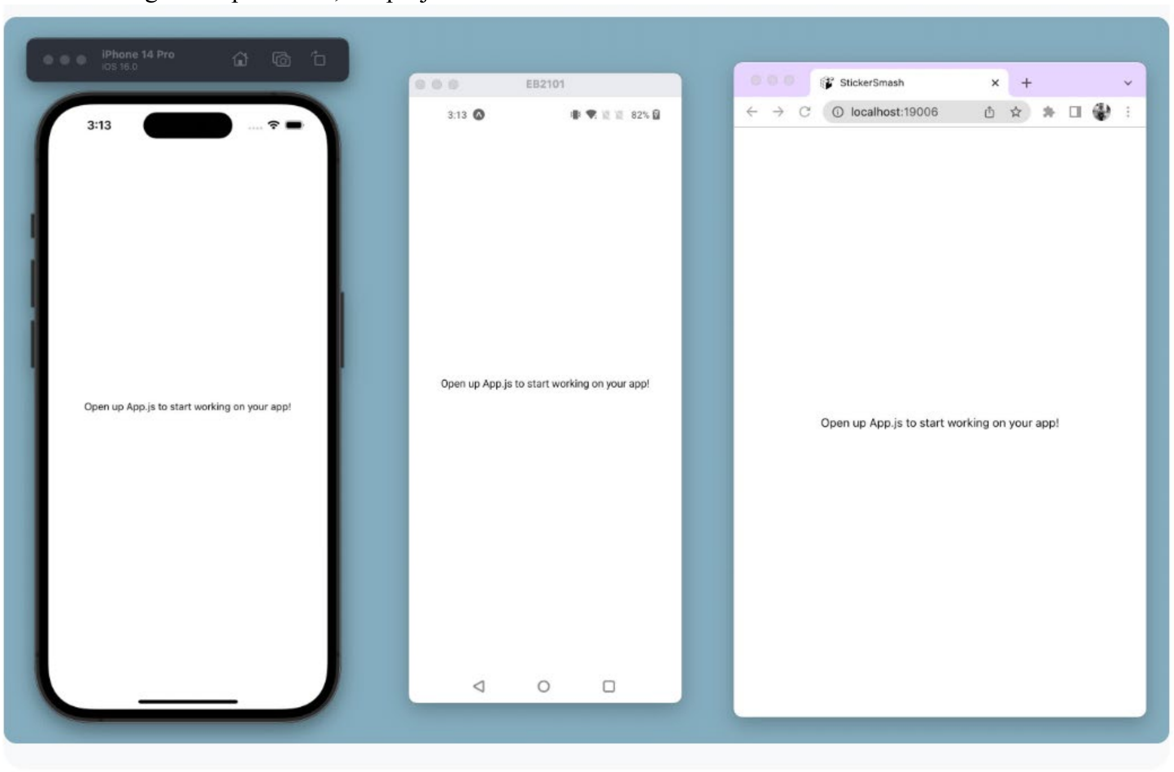
3. **Run the app on mobile and web**

npm start

Once the development server is running, the easiest way to launch the app is on a physical device with Expo Go. For more information, see Open app on a device.

To see the web app in action, press w in the terminal. It will open the web app in the default web browser.

Once it is running on all platforms, the project should look like this:


iPhone 14 Pro
iOS 16.0
3:13
Open up App.js to start working on your app!
EB2101
3:13
82%
Open up App.js to start working on your app!
StickerSmash
localhost:19006
Open up App.js to start working on your app!

The text displayed on the app's screen above can be found in the **App.js** file which is at the root of the project's directory. It is the entry point of the project and is executed when the development server starts.

When learning React Native, you create a lot of projects - at least we do here on Galaxies.dev!

To make it easier for you to get started, we have created a list of all React Native Expo templates that are up to date an useful to know.

React Native Expo Templates

There are many Expo Templates on Github but these are the currently maintained React Native Expo templates:

Blank

Blank (Typescript)

Navigation (Typescript)

Blank (Bare)

Let’s quickly go through them and see what they are about.

Blank
The blank template is the most basic template you can get. You can create it with the following command:

Command

```
npx create-expo-app --template blank
```

This template is using Javascript and has no navigation included.

As we prefer Typescript on Galaxies.dev, we would recommend using the Typescript version of this template instead.

Blank (Typescript)
This is the same template as before, but this time it is using Typescript:

Command

```
npx create-expo-app --template blank-typescript
```

Still, there is no navigation included so you would have to set up and configure everything.

Navigation (Typescript) - Recommended
This template is the most useful one in my opinion. It is using Typescript and has navigation included, which means it’s using the Expo Router:

Command

```
npx create-expo-app --template tabs
```

This is the template we are using the most in our React Native PRO courses.

Blank (Bare)
This template is using Javascript, has no navigation included but is using the bare workflow of Expo:

Command

```
npx create-expo-app --template bare-minimum
```

This means, you are not using the Expo Go app but manage your native dependencies yourself.

Core Components

Basic Components

Most apps will end up using one or more of these basic components.

View
The most fundamental component for building a UI.

Text
A component for displaying text.

Image
A component for displaying images.

TextInput
A component for inputting text into the app via a keyboard.

ScrollView
Provides a scrolling container that can host multiple components and views.

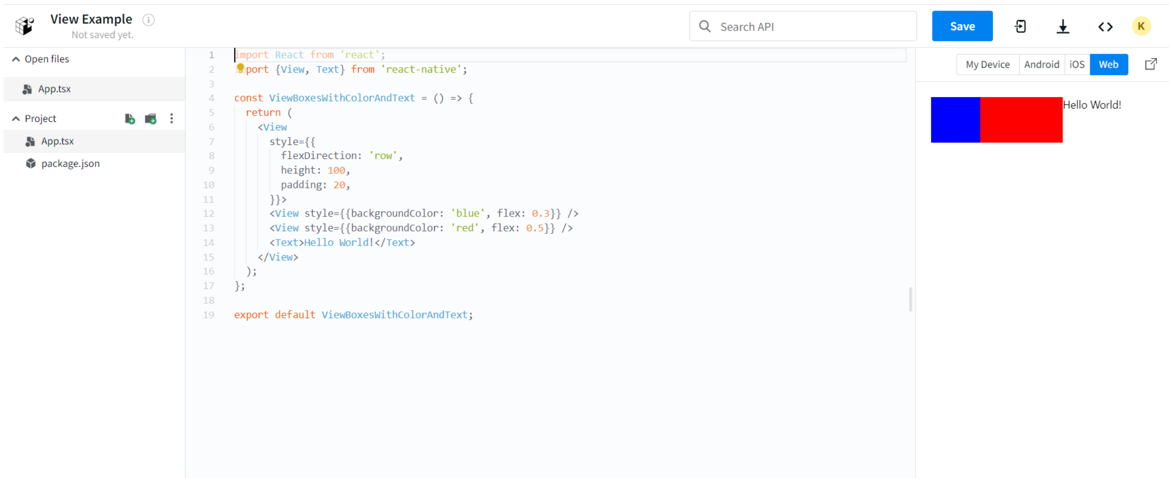
StyleSheet
Provides an abstraction layer similar to CSS stylesheets.

User Interface

These common user interface controls will render on any platform.

Button
A basic button component for handling touches that should render nicely on any platform.

Switch
Renders a boolean input.



```
import React from 'react';
import {View, Text} from 'react-native';

const ViewBoxesWithColorAndText = () => {
  return (
    <View
      style={{
        flexDirection: 'row',
        height: 100,
        padding: 20,
      }}>
      <View style={{backgroundColor: 'blue', flex: 0.3}} />
      <View style={{backgroundColor: 'red', flex: 0.5}} />
      <Text>Hello World!</Text>
    </View>
  );
};

export default ViewBoxesWithColorAndText;
```